НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

"На правах рукопису"
УДК_____

«До захисту допущено»
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2018р.

Магістерська дисертація

зі спеціальності 121 Інженерія програмного забезпечення
за спеціалізацією Програмне забезпечення розподілених систем
на тему ”Методи прискорення порівняння зображень”

Виконав: студент 6 курсу, групи _____
Жукова Олена Анатоліївна _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник к.т.н, доцент Верлань А. А. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Інститут/факультет Теплоенергетичний факультет
(повна назва)

Кафедра Кафедра автоматизації проектування енергетичних процесів і систем
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною (освітньо-науковою) програмою

Спеціальність (спеціалізація) 121 Інженерія програмного забезпечення
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (ініціали, прізвище)

«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Жукової Олени Анатоліївни
(прізвище, ім'я, по батькові)

1. Тема дисертації "Методи прискорення порівняння зображень"

науковий керівник дисертації кандидат технічних наук, доцент Верлань А.А. ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження процес порівняння зображень

4. Предмет дослідження Вихідні дані – для магістерської дисертації за освітньо-професійною програмою) алгоритми створення дескрипторів зображень, порівняння дескрипторів зображення та фактори, які впливають на швидкість порівняння зображення

5. Перелік завдань, які потрібно розробити
5.1. Провести дослідження існуючих програмних реалізацій детектора точок інтересу
5.2. Застосувати метод SURF для опису точок інтересу зображення.
5.3. Розробити математичну модель дескрипторів точок інтересу
5.4. Розробити програмну реалізацію дескрипторів точок інтересу
5.5. Розробити програмну реалізацію порівняння дескрипторів точок інтересу

6. Перелік графічного (ілюстративного) матеріалу Графіки, діаграми, таблиці, схеми, інтерфейси користувача

7. Орієнтовний перелік публікацій Загальна кількість запланованих статей для публікації – одна у фаховому виданні

8. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1.	Отримати завдання на МД	17.05.2018	
2.	Аналіз літературних даних з теми МД	01.06.2018-03.09.2018	
3.	Побудова математичних моделей	03.09.2018-21.09.2018	
4.	Проектування системи	21.09.2018 - 12.10.2018	
5.	Програмування та тестування	12.10.2018 - 21.11.2018	
6.	Підготовка МД та презентації для захисту	21.11.2018 - 26.11.2018	
7.	Предзахист МД та допуск до захисту	26.11.2018	
8.	Подання МД рецензенту. Отримання рецензії.	01.12.2018	
9.	Подання пакету документів по МД до захисту	10.12.2018	
10.	Захист МД в ЕК	19.12.2018	

Студент _____ Жукова О.А.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Верлань А.А.
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено наукового керівника

РЕФЕРАТ

Актуальність теми. Системи комп’ютерного зору набули неабиякої актуальності останнім десятиліттям. Задача знаходження відповідностей між двома зображеннями однієї сцени або об’єкту є частиною багатьох застосувань комп’ютерного зору, таких як реконструкція та реєстрація зображень, розпізнавання об‘єктів та багатьох інших.

Об’єктом дослідження є процес порівняння зображень.

Предметом дослідження є алгоритми створення дескрипторів зображень, порівняння дескрипторів зображення та фактори, які впливають на швидкість порівняння зображення.

Мета роботи: метою дослідження є розробка алгоритму порівняння зображень, який показує вищу швидкість порівняно з іншими аналогічними алгоритмами, при цьому зберігаючи точність співставлень.

Наукова новизна полягає у розробці методу описання характеристик зображення, який прискорює швидкість порівняння зображення, зберігаючи точність співставлень. Розроблений алгоритм який дозволяє зменшити розмір дескриптора з двовимірного масиву десяткових дробів до трьох бітових рядків.

Практична цінність полягає у прискоренні проведення порівнянь характеристик зображень, досягнутої завдяки зменшенню розміру даних необхідних для описання характеристик зображень, а також об’єму пам’яті для одного дескриптора, що особливо важливо для порівнянь великої кількості зображень, наприклад, при порівнянні відео.

Апробація роботи. Основні положення і результати роботи були представлені у публікації “Дескриптор точки інтересу у частотному діапазоні” у Міжнародному електронному науковому журналі Наука онлайн (№12, 2018 рік).

Ключові слова*: комп'ютерний зір, цифрові зображення, детектор ознак, дескриптор ознак, порівняння зображень.*

ABSTRACT

Relevance. Computer vision systems have gained a great deal of relevance in recent decades. The task of matching between two images of a single scene or object is part of many applications of computer vision, such as image reconstruction and registration, object recognition, and many others.

The object of research is the process of comparing images.

The subject of the study is algorithms for creating image descriptors, comparing descriptor images and factors that influence the speed of image comparison.

Purpose: The purpose of the study is to develop an algorithm for comparing images, which shows a higher speed compared to other similar algorithms, while maintaining the accuracy of the comparisons.

Scientific novelty consists in the development of a method for describing the characteristics of the image, which accelerates the speed of comparison of the image, while maintaining the accuracy of the comparisons. The algorithm is developed, which allows to reduce the size of the descriptor from a two-dimensional array of decimal fractions to three bits of lines.

The practical value is to accelerate the comparative performance of images achieved by reducing the size of data needed to describe the characteristics of images, as well as the amount of memory for one descriptor, which is especially important for comparing a large number of images, for example, when comparing video.

Publications. The main provisions and results of work were presented in the publication "Decoder of interest in the frequency range" in the International Electronic Science Magazine Science Online (No. 12, 2018).

Keywords: computer vision, digital image, interest points detector, features descriptor, image comparison, features matching.

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

DCT - дискретна косинусна трансформація

DFT - дискретна трансформація Фур'є

DoG (Difference of Gaussians) - різниця Гаусових фільтрів

LoG (Laplacian-of-Gaussian) - фільтр Лапласа, застосований до фільтру Гаусса

NMS (non-max-suppresion) - пригнічення не максимальних значень

SIFT (Scale-Invariant Feature Transform) - трансформація характеристик, незмінна за масштабування, метод виявлення ключових точок та створення дескрипторів характеристик, представлений у роботі "Distinctive image features from scale-invariant keypoints, cascade filter- ing approach" Девідом Лоу [1];

SURF (Speeded Up Robust Features - SURF) - прискорені надійні характеристики, метод виявлення ключових точок та створення дескрипторів характеристик, представлений у роботі "SURF: Speeded Up Robust Features" Гербертом Беєм [2];

ВСТУП

Сучасні застосування комп'ютерного зору широко використовуються для виконання багатьох практичних завдань, таких як реєстрація зображень, розпізнавання об'єктів, 3D реконструкція та інші. Розробка математичного обчислювального представлення зображення є передумовою для таких застосувань або алгоритмів. У багатьох існуючих моделях таке представлення створюється за допомогою наступних кроків:

1. виявлення точки інтересу, тобто виявлення характеристик зображення які є стійкими до шуму, помилок у виявленні та геометричних або фотометричних деформацій;
2. описання характеристики (дескриптор), тобто створення математичного описання, яке забезпечує можливість чисельного обрахування порівняння характеристик;
3. співставлення пари дескрипторів, засноване на оцінці їх схожості, наприклад, евклідова відстань між векторами дескрипторів.

Історично більшість детекторів ключових точок основані на першій та другій просторовій похідній. Найбільш значними представниками такого підходу є кутовий детектор Гарріса [3], який використовує матрицю других похідних та його стійкі до масштабування модифікації, наприклад, оператори Лапласа Гаусса, різниці Гауса SIFT [1] та швидкий детектор Гессе SURF [2]. Серед зазначених, швидкий детектор Гессе показує найкращі показники з продуктивності, у той же час, його результати є стійкими до зміни освітлення та зміщення.

Серед детекторів які не основані на алгоритмі Гарріса, можна виділити FAST [4], який є швидшим за SURF, але він не є стійким до масштабування. В результаті виявлення точки інтересу отримуємо пару координат та масштаб (x,y,s), які передаються як вхід до дескриптора характеристики.

Найбільш помітними реалізаціями дескрипторів характеристик є SIFT та SURF, які вважаються стандартами галузі для сучасних застосувань. Однак, розмір цих дескрипторів варіюється від 144 до 512 байт, що призводить до великих потреб у пам'яті та створює перешкоди для великомасштабних застосувань. Теоретично, дескриптор розміром 144 байти (1152-біт) може відрізнити 2^{1152} різних об'єкти. Це є набагато більше ніж потрібно у будь-якому практичному застосуванні. Більш того, дескриптори SIFT та SURF базуються на векторах з десятковими дробами, через що розрахунок відстані потребує більше часу.

Відповідно, бажаною є розробка такого дескриптора характеристик, який використовує тільки побітові операції або операції з цілими числами для розрахунку відстані та використовує помітно менший об’єм пам'яті. Розробка такого дескриптора може значно прискорити етап співставлення для великого масиву зображень.

Такий дескриптор створюється шляхом поділу площі навколо точки інтересу на області, конкретний метод поділу є компромісом між продуктивністю і виразність. Після чого, кожна область описується незалежно.

Для опису точок інтересу зображення виражається у частотному діапазоні, а саме дискретній косинусній трансформації (DCT).

На відміну від дескрипторів SIFT чи SURF, де відстань вираховується як евклідова відстань між векторами дескрипторів, для дескрипторів у частотному діапазоні відстань визначається як відстань Геммінга між двома послідовностями бітів. Таким чином відстань між двома дескрипторами може визначатися як вага Геммінга, що означає побітову операцію XOR з мірою складності O(log n) де n - є довжина бітового рядка.

Таким чином досягається значне збільшення швидкості порівняння дескрипторів зі збереженням точності описання.

1. ОСНОВНІ ПОНЯТТЯ

1.1. Типи зображень

Для даного дослідження важливо дати визначення типів зображень, які найчастіше використовуються у процесі порівняння зображень.

Зображення у відтінках сірого (зображення інтенсивності)

Дані зображення у відтінках сірого складаються з одного каналу, який виражає інтенсивність, яскравість або щільність зображення. Дані у таких зображеннях мають значення цілих чисел більше або рівних нулю, верхній ліміт визначають як 2^{k-1}. Наприклад, типове зображення у відтінках сірого, яке використовується у даному дослідженні, використовує $k = 8$ біт (1 байт), що відповідає діапазону $[0, 255]$, де 0 відповідає мінімальній яскравості (чорний), а 255 - максимальній яскравості (білий).

Бінарні зображення

Бінарні зображення - це особливий тип зображень, в яких значення пікселів може приймати лише одне з двох чисел 0 або 1, що відповідає білому або чорному кольору.

Кольорові зображення

Більшість кольорових зображень використовують модель з трьох кольорових каналів червоного, зеленого та синього (RGB), зазвичай використовують по 8 біт для представлення кольору в кожному з каналів.

Для даного дослідження важливо визначити ще й кольорову модель YUV, в якій колір представляється трьома компонентами — яскравість (Y) і два канали кольору (U і V). В даному дослідженні ця модель використовується для виділення лише каналу яскравості (Y).

Інтегральні зображення

Інтегральні зображення або таблиця сум площ (summed-area table) [5] є структурою даних та алгоритмом для ефективного розрахунку і зберігання сум

значень пікселів таким чином, що значення в будь-якій точці (x, y) цієї таблиці є сумою значень усіх пікселів зверху та зліва від цієї точки (x, y), включаючи значення в самій точці.

$$I(x, y) = i(x, y) + I(x, y - 1) + I(x - 1, y) - I(x - 1, y - 1) \quad (1.1)$$

Такі зображення зменшують складність інших алгоритмів у частині обрахування суми пікселів до константного значення $O(1)$.

1.2. Функція зображення

В основі усіх операцій з обробки цифрових зображень, що є предметом даного дослідження лежить представлення цифрового зображення як функції.

Припустимо, що безперервне зображення перетворено в двовимірний масив $I(u, v)$ - матрицю з *M* рядків і *N* стовпців, де (u, v) - дискретні координати. Для ясності позначень і більшої зручності ми будемо використовувати для цих координат цілочисельні значення: *u = 0, 1, 2, ..., M - 1* і *v = 0, 1, 2, ..., N - 1*, приймаючи за початок координат лівий верхній кут зображення, де $(u, v) = (0, 0)$.

У загальному випадку значення зображення в довільній координатної точці (u, v) позначається $I(u, v)$, де *u* і *v* - цілі числа. Область дійсної координатної площини, що охоплюється координатами зображення, називається просторовою областю, а *u* і *v* - просторовими змінними або просторовими координатами.

Таким чином, надалі цифрове зображення I визначається як двовимірна функція $\mathbb{N} \times \mathbb{N}$, яка співвідносить значення координат u, v зі значенням інтенсивності пікселя p.

$$I(u, v) = p \qquad u, v \in \mathbb{N} \tag{1.2}$$

Значення координат u, v належеть до діапазону $0 \le u \le M$ та $0 \le v \le N$ відповідно, де M - ширина, N - висота зображення. Значення пікселя p належить до діапазону значень $0 \le p \le 2^{k-1}$.

1.3. Похідна

В попередньому розділі дане визначення зображення I як функції, для вирішення задач в цьому дослідженні корисно надати визначення похідної від функції зображення, визначеної у рівнянні (1.1).

Спочатку розглянемо визначення для одновимірної функції, коли похідна від функції визначається як:

$$f'(x) = \frac{df}{dx}(x) \tag{1.3}$$

Похідна покаже додатні значення для збільшення інтенсивності і від'ємні - для зменшення. Однак у випадку дискретної функції, як зображення, необхідно застосувати максимальне наближення. Поскільки похідна по суті є мірою нахилу дотичної у точці x, для дискретної функції найближчим наближенням буде нахил лінії проведеної між двома найближчими точками.

Таким чином, для дискретної функції $f(u)$ маємо визначення похідної як:

$$f'(u) = \frac{df}{du}(u) \approx f(u+1) - f(u) \tag{1.4}$$

Відповідно похідною другого порядку буде:

$$f''(u) = \frac{d^2f}{du^2}(u) \approx f(u+1) - f(u-1) - 2f(u) \tag{1.5}$$

Для двовимірної функції зображення $I(u,v)$ похідна складається із часткових похідних для кожної змінної:

$$I'_u = \frac{dI}{du}(u) \quad \text{та} \quad I'_v = \frac{dI}{dv}(v) \tag{1.6}$$

Вектор похідних

$$\nabla I(u,v) = \begin{pmatrix} I'_u \\ I'_v \end{pmatrix} \tag{1.7}$$

називають градієнтом функції $I(u,v)$, величиною градієнта є

$$|\nabla I| = \sqrt{I'^2_u + I'^2_v} \tag{1.8}$$

Важливо відзначити, що величина градієнта не залежить від повороту зображення, ця властивість має велике значення для даного дослідження.

Похідні дискретної функції визначаються в термінах різниць. Ці різниці можна задати різними способами, однак ми будемо керуватися таким. Перша похідна:

(1) дорівнює нулю на областях з постійним рівнем яскравості;

(2) ненульова на початку і в кінці перепаду яскравості;

(3) ненульова на схилах яскравості.

Аналогічно друга похідна:

(1) дорівнює нулю на плоских ділянках;

(2) ненульова на початку і в кінці перепаду або схилу яскравості;

(3) дорівнює нулю на схилах постійної крутизни.

Так як ми оперуємо обмеженими чисельними значеннями, максимальне значення зміни яскравості також є кінцевим, а найкоротша відстань, на якому ця зміна може відбуватися, є відстань між сусідніми пікселями.

1.4. Згортка

Існують дві близькоспоріднені концепції, які при виконанні лінійної просторової фільтрації необхідно чітко розуміти. Одна з них - кореляція, а інша - згортка. Кореляція є процес руху маски фільтра по зображенню і обчислення суми творів значень елементів маски і значень пікселів, на які потрапляють відповідні елементи маски, для всіх точок зображення; в точності, як пояснювалося в попередньому розділі. Механізми згортки такі ж, але за винятком того, що попередньо маска фільтра повертається на 180 °.

Математична операція корреляції і згортки, тобто поєднання двох функцій з однаковою вимірністю для дискретної функції зображення $I(u,v)$ та функції $H(i,j)$ визначається наступним чином для корреляції:

$$I(u,v) \otimes H(i,j) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(u+i,\, j+v)H(i,j) \tag{1.9}$$

В операції згортки додавання елементів замінюється відніманням:

$$I(u,v) * H(i,j) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(u-i,\, j-v)H(i,j) \tag{1.10}$$

У літературі часто зустрічаються терміни “фільтр згортки” (convolution filter), “маска згортки”(convolution mask) або “ядро згортки” (convolution kernel), які означають операції просторових фільтрів, аналогічно вираз “згортати маску з зображенням” використовують для позначення процесу ковзання, сумування додатків коефіцієнтів фільтра та значень функції зображення. Схематично цю операцію зображено на рис.1.1.

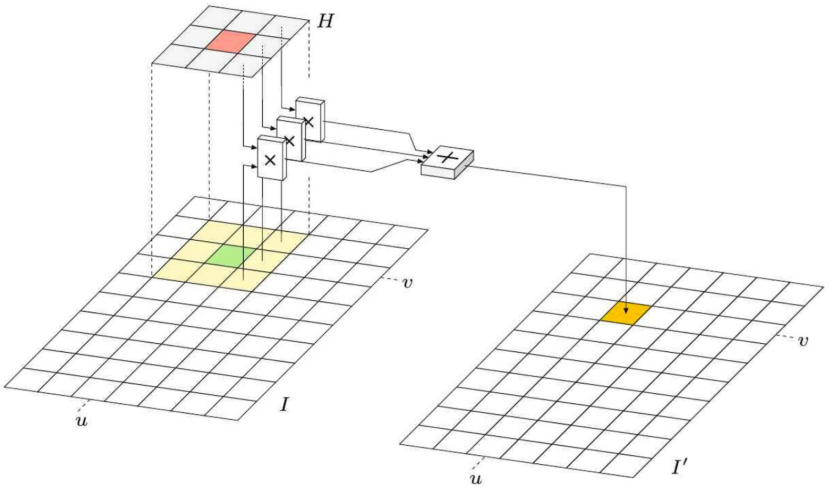

Рисунок 1.1. Принцип роботи лінійного фільтру: ядро фільтру H пересувають по площі зображення I таким чином, що точка $H(0,0)$ співпадає з точкою $I(u,v)$; усі коефіцієнти фільтра $H(i,j)$ множаться на відповідний елемент $I(u+i,v+j)$ результат додається і переноситься на відповідну позицію нового зображення $I'(u,v)$;

Розглянемо декілька типів фільтрів, що часто використовують в процесі порівняння зображень.

Фільтри згладжування

Фільтри в результаті яких отримуємо згладжене зображення складаються виключно з додатних коефіцієнтів.

Серед таких фільтрів виділимо три варіації: прямокутний фільтр (box filter), фільтр Гауса та фільтр Лапласа.

Найпростішим є прямокутний фільтр (box filter), коефіцієнти якого рівномірно розподілені навколо центру. Його матриця виглядає наступним чином:

$$H = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{1.11}$$

3D графік прямокутного фільтру нагадує коробку (рис. 1.2.)

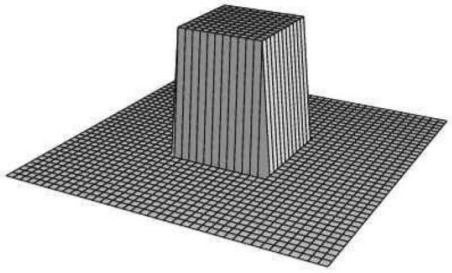

Рисунок 1.2. - Прямокутний фільтр (box filter)

Фільтр Гауса

Цей тип фільтра для визначення коефіцієнтів використовує двовимірну функцію Гауса, яка в базовому вигляді визначається як:

$$h(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{1.12}$$

Де σ - стандартне відхилення, x, y - координати фільтра

Матрицю такого фільтра можна визначити як:

$$G = \begin{pmatrix} 0 & 1 & 2 & 1 & 0 \\ 1 & 3 & 5 & 3 & 1 \\ 2 & 5 & 9 & 5 & 2 \\ 1 & 3 & 5 & 3 & 1 \\ 0 & 1 & 2 & 1 & 0 \end{pmatrix} \tag{1.13}$$

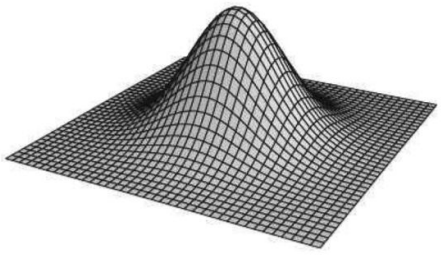

Рисунок 1.3. - Фільтр Гаусса

Фільтри в яких коефіцієнти від’ємні, називають фільтрами різниці, найрозповсюдженіший з яких фільтр Лапласа. Розрахунки з таким фільтром можна визначити як різницю двох сум: зважену суму всіх коефіцієнтів з

додатними значеннями та зважену суму всіх коефіцієнтів з від’ємними значеннями за такою формулою:

$$I'(u,v) = \sum_{(i,j)\in R^+} I(u+i,\ v+j)\cdot|H(i,j)| - \sum_{(i,j)\in R^-} I(u+i,\ v+j)\cdot|H(i,j)| \quad (1.14)$$

Наприклад для фільтра Лапласа з такою матрицею:

$$L = \begin{pmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{pmatrix} \quad (1.15)$$

Розраховуємо різницю між центральним пікселем (з вагою 16) та зваженою сумою сусідніх пікселів (з вагами -1 чи -2).

Графік такого фільтру схожий на сомбреро, через що його ще називають фільтр “сомбреро” ("Mexican hat" filter) рис 1.4.

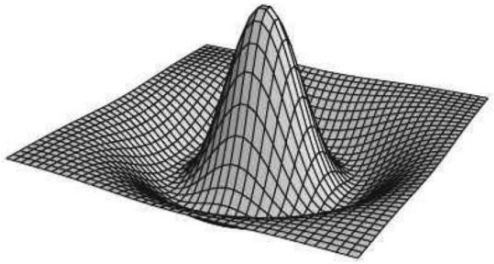

Рисунок 1.4. - Фільтр Лапласа

1.5. Точка інтересу

Центральним поняттям всього процесу є поняття точки інтересу. Точкою інтересу є точка на зображенні, яка має точно визначені координати на зображенні та пов’язана зі значною зміною однієї або декількох характеристик зображення (найчастіше, яскравості). Точки інтересу знаходяться у координатах пікслів перепаду.

Пікселі перепаду - це ті пікселі зображення, в яких функція яскравості різко змінюється, а перепад (або ділянку перепаду) - це чіткий безліч пікселів перепаду. Детектори перепадів представляють собою методи локальної обробки зображення, спрямовані на виявлення пікселів перепаду. Лінію можна розглядати як ділянку перепаду, у якого яскравість фону по обидва боки від лінії або значно вище, або значно нижче, ніж яскравість пікселів лінії.

Зміни яскравості між суцільними об'єктами і фоном уздовж лінії сканування демонструють два види перепадів: похилий перепад і ступінчастий перепад. Як йдеться нижче, зміни яскравості на тонких об'єктах, таких як лінія, часто називають трикутним імпульсом.

Виявлення точки інтересу є базою для усіх інших операцій з порівняння зображень.

1.6. Просторова частота

Зображення можна виразити не тільки в діапазоні рівнів сірого, але й у частотному діапазоні, для даного дослідження корисно надати визначення просторової частоти.

Загальним міждисциплінарним визначенням просторової частоти є мірою повторюваності значень (циклу) на одиницю часу. У міжнародному стандарті мір одиницею цього типу частоти є цикл на метр. Це поняття базується на дослідженнях Кембела та Робсона сприйняття людиною простору, завданням яких було винайти спосіб вимірювання сигналу до зорової кори головного мозку. Центральним поняттям цієї теорії стали гратки, що повторюють чорну та білу лінії. Таким чином період для просторової частоти стала пара білої та чорної лінії, а одиницею простору є кут зору (кут який відтинає зображення на сітчатці ока), позначений θ на рис 1.5.

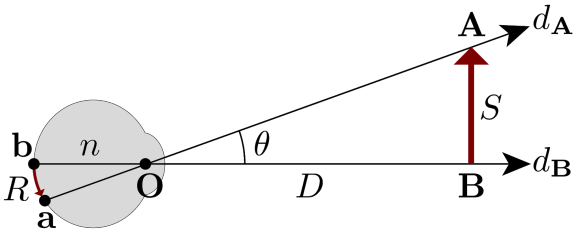

Рисунок 1.5. - Кут зору (Perceived visual angle)

O - центр зіниці ока

D - відстань від точки O

A, B - межі об‘єкта спостереження

S - розмір об’єкта спостереження

a, b - межі проекції об’єкта спостереження на сітчатку ока

R - довжина дуги, проекції об’єкта спостереження на сітчатці ока

θ - кут зору.

Так, одна пара ліній на один градус кута зору є одиницею просторової частоти.

Повторюваність граток становлять синусоїдну функцію просторової частоти.

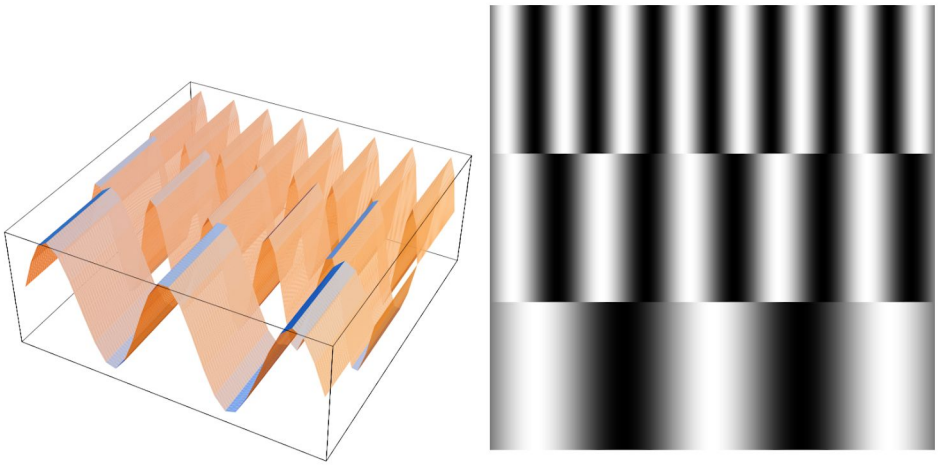

Рисунок 1.6. - Приклад трьох різних просторових частот

За дослідженнями Кембела та Робсона, будь яке зображення, яке сприймає людина можна виразити через частоту, контраст, напрямок та зміщення граток.

2. ІСНУЮЧІ АЛГОРИТМИ ПОРІВНЯННЯ ЗОБРАЖЕНЬ

Алгоритм порівняння зображень складається з трьох етапів:

1. Виявлення точки інтересу

2. Описання точки інтересу

3. Співставлення описів точок інтересу.

Виявлення точки інтересу є виявленням місця найбільш значної зміни інтенсивності зображення і є кінцевим результатом процесу з декількох кроків: виявлення контурів, виявлення кутів, виявлення точок.

Теорія виявлення контурів об’єктів основана на твердженні, що об’єкти на зображенні відрізняються від фону рівнем інтенсивності. Таким чином, щоб визначити контур об’єкта необхідно вирахувати зміни інтенсивності по всій площі зображення. Таку задачу можна вирішити за допомогою похідних від функції зображення, які визначені у розділі 1.3. або фільтрів різниці.

У таблиці 1. Наведені відомі реалізації фільтрів різниці.

Таблиця 1. - Реалізації фільтрів різниці

Назва	Ядро згортки	
	горизонтальний напрямок	Вертикальний напрямок
Базовий фільтр різниці	0 0 0 -1 0 1 0 0 0	0 -1 0 0 0 0 0 1 0
Прюїта	-1 0 1 -1 0 1 -1 0 1	-1 -1 -1 0 0 0 1 1 1

Собеля

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

Робертса

0	1
-1	0

-1	0
0	1

Лапласа

1	1	1
1	-8	1
1	1	1

Аналізуючи зміни інтенсивності зображення можна зробити висновок, що незначна зміна інтенсивності є характерною для ділянок зображення де немає контурів, такі ділянки називають рівнинами, помітна зміна інтенсивності в одному напрямку (горизонтальному або вертикальному) виявляє край об’єкта, а зміна інтенсивності в обох напрямках виявляє кут.

2.1. Кутовий детектор Гарріса

Найбільш вживаним з кутових детекторів є кутовий детектор Гарріса [3], в основі якого градієнт з часткових похідних функції зображення $I'(u,v)$, відповідно до рівняння (1.3).

Для кожної точки (u,v) розраховуються три значення:

$$A(u,v) = {I'_u}^2$$
$$B(u,v) = {I'_v}^2$$
$$C(u,v) = I'_u \cdot I'_v$$

Які складають елементи матриці $M(u,v)$:

$$M = \begin{pmatrix} A & C \\ C & B \end{pmatrix}$$

Для кожного з цих значень застосовується фільтр Гауса G див. рівняння (1.13)

$$M^* = \begin{pmatrix} A*G & C*G \\ C*G & B*G \end{pmatrix} = \begin{pmatrix} A^* & C^* \\ C^* & B^* \end{pmatrix}$$

Наступним кроком вираховують власні значення матриці M^*

$$\lambda_{1,2} = \frac{\operatorname{trace}(M^*)}{2} \pm \sqrt{\frac{\operatorname{trace}(M^*)}{2} - \det(M^*)}$$
$$= \frac{1}{2}(A^* + B^* \pm \sqrt{A^{*2} - 2A^*B^* + B^{*2} + 4C^{*2}})$$

З попереднього рівняння маємо:

$$\lambda_1 - \lambda_2 = 2 \cdot \sqrt{0.25 \cdot (\operatorname{trace}(M^*))^2 - \det(M^*)}$$

У місці де є кут значення λ_1, λ_2 мінімальні.

Для того, щоб не вираховувати власні значення матриці (та квадратний корінь) кутовий детектор Гарріса визначає функцію:

$$Q(u,v) = \det(M^*) - \alpha(\operatorname{trace}(M^*))^2 \qquad (2.1)$$
$$= A^*B^* - {C^*}^2 - \alpha(A^* + B^*)^2$$

Де параметр α означає чутливість детектора, який на практиці визначають значенням від 0.04 до 0.06. Чим більше значення α, тим менш чутливим буде детектор, тобто менше кутів буде виявлено.

Функцію $Q(u,v)$ називають функцією кутової відповіді (corner response function), яка повертає найбільше значення для точок в області кутів.

Завершальною стадією виявлення ключових точок шляхом відкидання точок, що знаходяться в області кутів (non maximum suppression).

Точка (u,v) на зображенні вибирається як потенційний кандидат на ключову точку, якщо $Q(u,v) > T$, де T - заздалегідь визначений поріг, усі кандидати додаються до списку, який сортують у порядку зменшення значення $Q(u,v)$, для того, щоб відкинути точки, що не є ключовими, відкидають точки значення $Q(u,v)$ яких менші порівняно з сусідніми.

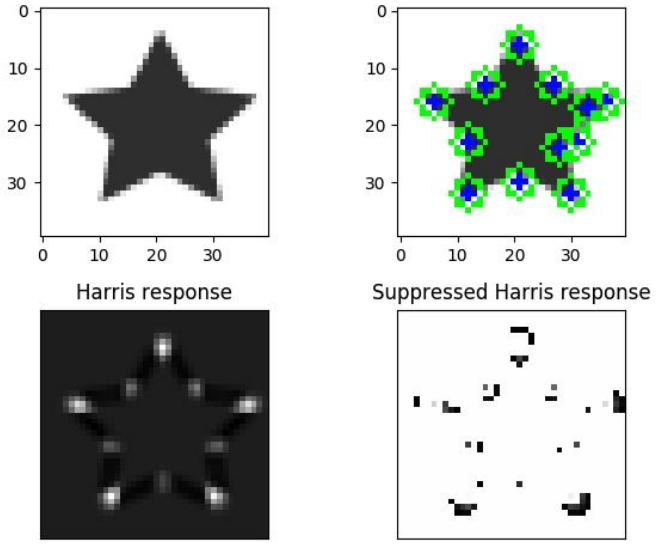

Рисунок 2.1. Приклад роботи кутового детектору Гарріса.

2.2. Трансформація характеристик, незмінна за масштабування (SIFT)

Техніка виявлення ключових точок, що називається трансформацією характеристик, незмінною від масштабу (Scale-Invariant Feature Transform - SIFT) представлена Девідом Лоу покращила результати роботи детектора Гарріса, зробивши його результати стійкими до змін масштабу. SIFT застосовує концепцію “простору масштабів”, для того, щоб захопити визначальні характеристики та декількох масштабах або рівнях роздільної здатності зображення.

Основні кроки для виявлення ключових точок за допомогою SIFT є наступними:

1. Визначити екстремуми за допомогою фільтрів Лапласа-Гаусса (LoG) для визначення потенційних точок інтересу;
2. Уточнення ключових точок;
3. Визначення напрямку характеристики через визначення градієнта області навколо ключової точки;

У підході SIFT виявлення точки інтересу основане на фільтри Лапласа-Гаусса (LoG), який виявляє світлі плями на темному тлі та навпаки. Для того, щоб виявити точку на різних рівнях масштабу, створюється простір масштабів, шляхом рекурсивного згладжування зображення послідовністю невеликих фільтрів Гаусса.

Різниця між зображеннями у на сусідніх рівнях масштабу використовується для розрахунку апроксимації фільтра LoG.

Фільтр Лапласа для двовимірної функції визначається як сума похідних другого порядку:

$$(\nabla^2 f)(x,y) = \frac{\delta^2 f}{\delta x^2}(x,y) + \frac{\delta^2 f}{\delta y^2}(x,y) \tag{2.2}$$

Слід зауважити, що на відміну від інших фільтрів в результаті застосування LoG отримаємо одне скалярне значення.

Якщо застосувати фільтр Лапласа до фільтра Гаусса (2.3)

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$$

отримаємо LoG:

$$\begin{aligned} L_\sigma(x,y) = (\nabla^2 G_\sigma)(x,y) &= \frac{\delta^2 G_\sigma}{\delta x^2}(x,y) + \frac{\delta^2 G_\sigma}{\delta y^2}(x,y) \\ &= \frac{(x^2-\sigma^2)}{2\pi\sigma^6} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}} + \frac{(y^2-\sigma^2)}{2\pi\sigma^6} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}} \\ &= \frac{1}{\pi\sigma^4} \cdot \left(\frac{x^2+y^2-2\sigma^2}{2\sigma^2}\right) \cdot e^{-\frac{x^2+y^2}{2\sigma^2}} \end{aligned} \tag{2.4}$$

Коли LoG використовують як ядро лінійного фільтра, максимальні значення він набуває для темних плям на світлому тлі, а для темних плям на світлому тлі використовують від'ємний LoG, а обидва типи плям можуть бути виявлені шляхом використання абсолютних значень результату фільтру.

Зазвичай LoG реалізують через апроксимацію за допомогою різниці різниці двох фільтрів Гаусса DoG з шириною σ та $k\sigma$:

$$L_\sigma(x,y) \approx \lambda \cdot [\underbrace{G_{k\sigma}(x,y) - G_\sigma(x,y)}_{=D_{\sigma,k}(x,y)}] \tag{2.5}$$

Множник λ контролює величину DoG, він залежить від співвідношення k та ширини σ

$$\lambda = \frac{2k^2}{\sigma^2 \cdot (k^2-1)} \tag{2.6}$$

Для того, що співвіднести виявлені структури на різних масштабах необхідно представити зображення на різних рівнях масштабу. Простір масштабів зображення додає масштаб як третій вимір для функції зображення.

Для кожної октави в просторі масштабі в початковий зображення повторювана згортається з фільтром Гаусса для того щоб створити набір у просторі масштабів як зображено зліва. Різниця сусідніх Гаусових зображень зображена справа. Після кожної октави гаусова зображення зменшують вдвоє та процес повторюється див. рис 2.2.

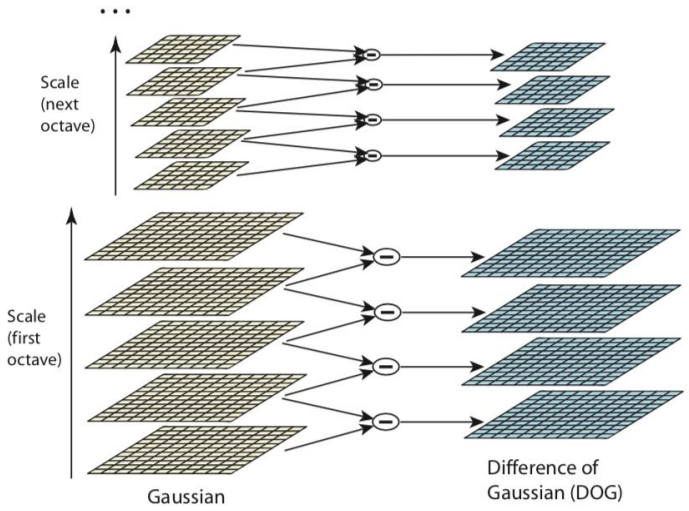

Рисунок 2.2. - побудова простору масштабів

Для того щоб виявити локальні максимуми і мінімуми функції кожна точка з вибірки порівнюється до її восьми сусідніх точок на поточному зображенні та дев'яти точок на масштабі зверху та знизу, як зображено на рис. 2.3.

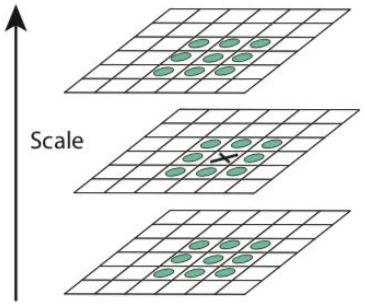

Рисунок 2.3. - Максимум та мінімум функції DoG на зображенні виявляється шляхом порівняння пікселю (відзначено хрестом) до його 26 сусідів у області розміром 3*3 на поточному та сусідніх масштабах (позначені кружечком).

Консистентна орієнтація призначається для кожної ключової точки на основі локальних характеристик зображення. Опис точки може бути виражений

відносно до цієї орієнтації і таким чином досягається стійкість до поворотів зображення. Для кожного зображення з вибірки, $L(x,y)$ у цьому масштабі величина градієнту $m(x,y)$ та орієнтація $\theta(x,y)$ вираховується шляхом різниці пікселей:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (2.7)$$

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right) \quad (2.8)$$

Гістограма орієнтації формується з орієнтації градієнтів навколо ключової точки. Ця гістограма має 36 стовпчиків, що покривають 360° орієнтацій. Кожний зразок додається до гістограми та помножений на величину градієнта. Піки в гістограмі орієнтацій відповідають домінантною напрямку локального градієнту. Визначається найвищий пік у гістограмі, а потім інші локальні піки, які мають висоту не менше 80% від найвищого піку, вони використовуються для того щоб створити ключову точку з такою орієнтацією. Таким чином, для локації з декількома піками схожої величини буде створено декілька ключових точок з однаковими координатами та масштабом але різними орієнтаціями. Близько 15% точок мають декілька орієнтацій але це значно впливає на стабільність порівняння.

Наступним кроком вираховуються дескриптори області зображення навколо ключової точки, які мають бути особливого виразними і в той же час залишатися стійкими до змін в освітленні та точки огляду.

Для визначення дескриптора ключової точки спочатку обраховується величина градієнтів та орієнтація у кожному зображенні для регіону навколо ключової точки як показано зліва на рисунку 2.4. Значення зважені на фільтр Гаусса та позначаються обведений колом. В результаті, складається одне значення яке описує регіон розміром 4*4 пікселі, як показано справа на рисунку 2.4., де довжина кожної стрілки відповідає сумі градієнтів цього напрямку в

цьому регіоні. Ця схема показує дескриптор розміром 2*2 розрахований з набору зразків розміром 8*8.

Порівняння дескрипторів здійснюється шляхом розрахунку Евклідової відстані між двома дескрипторами.

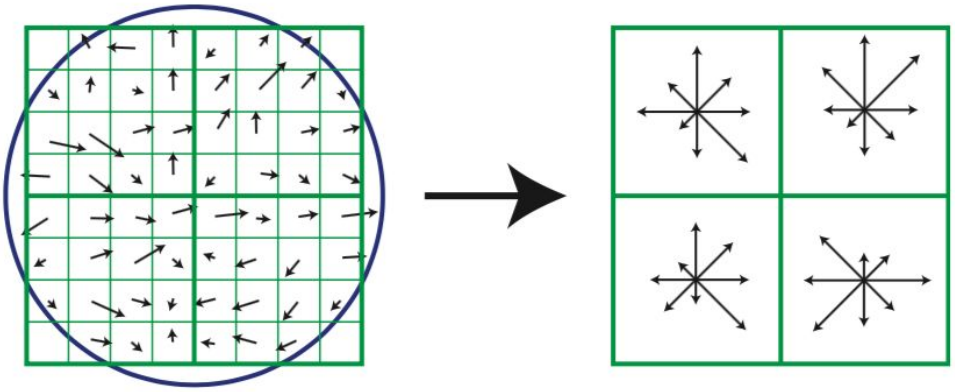

Рисунок 2.4. - визначення дескриптора ключової точки

2.3. Прискорені надійні характеристики (Speeded Up Robust Features - SURF)

Виявлення ключових точок в алгоритмі прискорених надійних характеристик (Speeded Up Robust Features - SURF) [2] базується на матриці других похідних Гессе, так само як і попередня техніка, але використовує базові наближення. Алгоритм використовує інтегральні зображення (див. рівняння 1.1) для того, щоб зменшити час для розрахунків, саме тому його називають “швидким детектором Гессе”.

Алгоритм заснований на матриці Гесса через швидкість її розрахунку та точність, однак замість того, щоб використовувати різні міри для вибору координат та масштабу, як це робилося в попередньому підході, даний алгоритм використовує детермінант матриці Гессе для обох показників. Маючи точку $p = (x, y)$ на зображенні I, матриця Гессе $H(p, \sigma)$ у точці p та масштабі σ визначається наступним чином:

$$H(p,\sigma) = \begin{pmatrix} L_{xx}(p,\sigma) & L_{xy}(p,\sigma) \\ L_{xx}(p,\sigma) & L_{yy}(p,\sigma) \end{pmatrix} \tag{2.9}$$

де $L_{xx}(p, \sigma)$ це згортка похідної другого порядку функції Гаусса (рівняння 1.12) з зображенням I у точці p так само і для $L_{xy}(p, \sigma)$ та $L_{yy}(p, \sigma)$.

Поскільки фільтр Гаусса є не ідеальним для дискретних задач, SURF застосовує наближення до фільтра Гаусса за допомогою прямокутних фільтрів.

Простір масштабів виражається у виді пірамід зображень, як і в попередньому випадку, однак завдяки застосуванню прямокутних фільтрів, замість зменшення розміру зображення збільшується розмір фільтру таким чином масштаб розраховується за допомогою фільтрів розміром 9*9, 15*15, 21*21, 27*27.

3. РОЗРОБКА АЛГОРИТМУ ПОРІВНЯННЯ ЗОБРАЖЕНЬ

Метою даного дослідження є створення дескриптора характеристик, який би точно та виразно описував характеристики зображення, при цьому час на його обчислення та порівняння був би помітно менший за інші реалізації.

Будь-яка операція порівняння зображень починається з етапу виявлення ключових точок, зважаючи на те, що цей етап не є предметом даного дослідження, для виявлення ключових точок на зображенні вирішено використати детектор SURF.

За опублікованими порівняннями детекторів [6], кількість ключових точок виявлених усіма детекторами є приблизно однаковою, але порівняння показують що “швидкий детектор Гессе” SURF є в 3 рази швидшим від SIFT та в п'ять разів швидше від детектора Лапласа-Гессе, при цьому повторюваність виявлення точки є на одному рівні з іншими. На підставі вказаних порівнянь, прийнято рішення про використання детектору SURF для стадії виявлення ключових точок.

Найбільш помітними реалізаціями дескрипторів характеристик є SIFT та SURF, які вважаються стандартами галузі для сучасних застосувань. Однак, розмір цих дескрипторів варіюється від 144 до 512 байт, що призводить до великих потреб у пам'яті та створює перешкоди для великомасштабних застосувань. Теоретично, дескриптор розміром 144 байти (1152-біт) може відрізнити 2^{1152} різних об'єкти. Це є набагато більше ніж потрібно у будь-якому практичному застосуванні. Більш того, дескриптори SIFT та SURF базуються на векторах з десятковими дробами, через що розрахунок відстані потребує більше часу.

3.1. Дискретні частотні перетворення для описання характеристик зображення

Перетворення Фур'є, що відноситься до предмету розгляду, полягає, по суті, в тому, що будь-яка періодична функція може бути представлена у вигляді суми синусів і / або косинусів різних частот, помножених на деякі коефіцієнти (тепер ця сума має назву ряд Фур'є). Не має значення, наскільки складною є функція; якщо тільки вона періодична і задовольняє математичним умовами, ця функція може бути представлена у вигляді вищевказаної суми.

Зображення розглядається як сукупність просторових хвиль, для яких осі X і Y проводяться паралельно поздовжньої і поперечної осях картинки, а по осі Z відкладається значення кольору відповідного пікселя зображення. Дискретне перетворення Фур'є (ДПФ) дозволяє переходити від просторового уявлення зображення до його спектрального представлення і назад. Формула дискретне перетворення Фур'є для цифрового зображення *f(x,y)* розміром *M*N:*

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-i2\pi\left(\frac{ux+vy}{MN}\right)} \tag{3.1}$$

Кожен елемент Фур'є-образу *F (u, v)* містить всі відліки функції *f (x, y)*, помножені на значення експоненціальних членів. Тому зазвичай, за винятком тривіальних випадків, неможливо встановити пряму відповідність між характерними деталями зображення і його Фур'є-образу. Однак деякі загальні твердження щодо взаємозв'язку частотних складових Фур'є-образу і просторових особливостей зображення можуть бути зроблені. Наприклад, оскільки частота прямо пов'язана зі швидкістю просторових змін сигналу, то інтуїтивно зрозуміло, що частоти в Фур'є-перетворення пов'язані з варіацією яскравості зображення.

Найбільш повільно змінюється (постійна) частотна складова (u = v = 0) пропорційна середньої яскравості зображення. Низькі частоти, що відповідають точкам поблизу початку координат Фур'є-перетворення, відповідають повільно змінюваним компонентам зображення. На зображенні кімнати, наприклад, вони

можуть відповідати плавним змінам яскравості стін і підлоги. По мірі віддалення від початку координат вищі частоти починають відповідати все більшій і більшій швидкості змін яскравості, які по-суті є межами об'єктів і інші деталі зображення, що характеризуються різкими змінами яскравості.

Низькі частоти в перетворенні пов'язані з областями повільно змінюється яскравості, такими як стіни кімнати або безхмарне небо на зображенні пейзажу. Високі частоти, навпаки, викликаються різкими переходами яскравості, такими як контури і шум.

Компонентами Фур'є-перетворення є амплітуда перетворення (Фур'є-спектр) і фаза. Візуальний аналіз фазової компоненти, як правило, має мало сенсу. Зате спектр дає деякі корисні вказівки на загальні характеристики вихідного зображення.

Таким чином, можна очікувати, що фільтр, що пригнічує високі частоти і пропускає низькі і відповідно званий фільтром низьких частот, буде розмивати зображення. В цей же час фільтр з протилежними характеристиками (званий фільтром високих частот) буде підвищувати контраст різких деталей, але призведе до зниження загального контрасту на зображенні.

Двовимірне ДПФ від $f(x, y)$ може бути отримано обчисленням одновимірних перетворень по кожному рядку f (x, y), а потім обчисленням одновимірних перетворень по кожному колонку отриманого проміжного результату. Це є важливим спрощенням, оскільки в кожен момент часу нам потрібно мати справу лише з однією змінною. Аналогічний підхід можна застосувати і до обчислення двовимірного зворотного ДПФ за допомогою одновимірного зворотного ДПФ.

Робота в частотної області буде не дуже зручною, якщо обчислювати операції прямим чином. На виконання цих перетворень потрібно близько $(MN)^2$ операцій множення і додавання. Для зображень середнього розміру (скажімо, 1024 × 1024 пікселів) це означає необхідність виконання близько 10^{12} множень і складань тільки для обчислення ДПФ, що не враховує необхідності обчислення

експоненти, яка може бути обчислена один раз і поміщена в таблицю. Такий обсяг обчислень скрутний навіть для суперкомп'ютера. Без швидкого перетворення Фур'є (ШПФ), яке скорочує число необхідних обчислень до порядку $MN\ log_2\ MN$ операцій множення і додавання, можна з упевненістю сказати, що матеріал, викладений в цій главі, мало придатний до практичного використання. Скорочення обчислень, що забезпечується БПФ, є вражаючим. Так, обчислення двовимірного ШПФ зображення розмірами 1024 × 1024 пікселів зажадає всього близько 2^{107} операцій множення і додавання, що є значним скороченням в порівнянні з початковим числом 1012, зазначеним вище.

Для мети цього дослідження використаємо двовимірне дискретно-косинусное перетворення, так як воно дозволяє позбутися уявної компоненти перетворення Фур'є . Двовимірне дискретно-косинусное перетворення зображень частіше використовують для компресії зображень у форматі JPEG. Його здатність зберігати максимум інформації у компактному розмірі відповідає меті даного дослідження по розробці швидкого і компактного дескриптора характеристик зображення.

Двовимірне дискретно-косинусное перетворення матриці L_{xy} з розмірами MxN реалізується відповідно до наступного виразу:

$$F(i,j) = C(i)C(j) \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} L_{xy} cos \left[\frac{(2x+1)i\pi}{2M} \right] cos \left[\frac{(2y+1)j\pi}{2N} \right] \tag{3.2}$$

$$C_i, C_j = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } i,j = 0 \\ 1 & \text{otherwise} \end{cases}$$

Значення F_{ij} називають коефіцієнтами дискретно-косинусного перетворення матриці L.

Коефіцієнти дискретно-косинусного перетворення можна розглядати як ваги при кожній базової функції. Наприклад, для матриці з розміром 8x8 елементів існує 64 базові функції, що продемонстровані на рис 3.1.

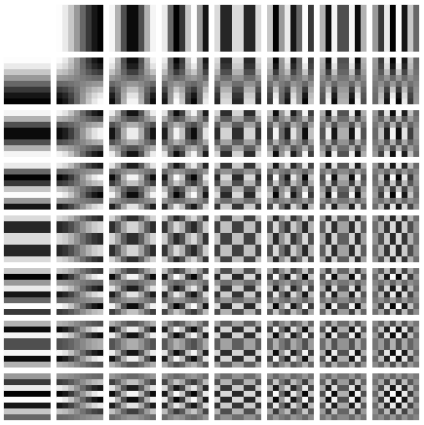

Рисунок 3.1. - базові функції дискретного косинусного перетворення

DCT найчастіше зустрічається в практичних додатках завдяки властивості “ущільнення енергії”. Це властивість означає, що максимальна кількість інформації укладено в перших коефіцієнтах.

3.2. Розробка дескриптора характеристик зображення

Дескриптор характеристик є набором цифр які ми можемо порівняти з дескрипторами на іншому зображенні.

Дескриптор характеристик створюється шляхом поділу площі навколо точки інтересу на області. Після чого, кожна область описується незалежно.

Кожна з областей, що описується поділяється на рівномірну ортогональну сітку, для прикладу візьмемо сітку розміром 8x8 комірок. Після чого вираховується середня освітленість для кожної з комірок, таким чином отримуємо 8x8 матрицю освітленості L_{xy}. Як необов'язковий крок, можна збільшити продуктивність шляхом завдяки інтегральним зображенням [7], що зробить розрахунок середньої освітленості операцією з константною складністю O(1). Наступним кроком матриця освітленості переводиться у частотний діапазон, шляхом дискретного перетворення Фур'є, точніше його різновиду, дискретного косинусного перетворення DCT [7].

Так як, DCT є достатньо дорогою операцією у сенсі продуктивності, доцільно застосувати DCT з цілими числами, замість DCT з десятковими дробами. В результаті цієї операції отримаємо 8x8 матрицю у частотному діапазоні F_{ij}.

$$F(i,j) = \frac{1}{4}C(i)C(j)\sum_{x=0}^{7}\sum_{y=0}^{7}L_{xy}cos\left[\frac{(2x+1)i\pi}{16}\right]cos\left[\frac{(2y+1)j\pi}{16}\right] \tag{3.3}$$

$$C_i, C_j = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } i,j = 0 \\ 1 & \text{otherwise} \end{cases}$$

В результаті перетворення зображення з просторового у частотне відображення ми отримаємо 64 числа, кожне з яких описує незалежну особливість зображення на відміну від просторового відображення в якому жоден піксель не може описати зображення або його область в цілому. Наприклад, коефіцієнт F_{00} відповідає середній освітленості всієї площі, F_{10} характеризує градієнт по осі ординат, аналогічно F_{01} відповідає градієнту по осі абсцис, а F_{22} відповідає

світлій або темній ділянці у центрі зображення (залежно від знаку + або -).

Так як коефіцієнт F_{00} відповідає середній освітленості, він може бути відкинутий для того щоб зробити дескриптор стійким до змін в освітленні. Після чого, можна зробити дескриптор стійким до змін контрасту шляхом нормалізації решти матриці

$$f_{ij} = \frac{F_{ij}}{\sum_{i+j\neq 0} |F_{ij}|} \tag{3.4}$$

Для того, щоб покращити продуктивність, нормалізацію можна обмежити обмежити лише цілими числами

$$f_{ij} = \frac{CF_{ij}}{\sum_{i+j\neq 0} |F_{ij}|} \tag{3.5}$$

де C є довільною достатньо великою константою наприклад 2^{16}.

Після нормалізації, кожен з коефіцієнтів матриці f_{ij} показує важливість характеристики відповідно до цього коефіцієнту у порівнянні з усіма іншими характеристиками. Таким чином, візуально значні характеристики будуть мати значні абсолютні значення їх відповідних коефіцієнтів.

Доцільно відкинути частину f_{ij} з високими частотами, так як високі частоти рідко є важливими та часто відповідають шуму на зображенні. Також, корисно зробити дескриптор якомога компактнішим. Завдяки візуальній незалежності характеристик, що описують коефіцієнти матриці усі інші коефіцієнти можуть зберігатися у вигляді лише одного біту в якому одиниця буде означати велике абсолютне значення цього коефіцієнту, а 0 - означатиме, що коефіцієнт f_{ij} близький до нуля. Для того, щоб провести таку класифікацію кожне значення $|f_{ij}|$ має бути порівняно з визначеним заздалегідь порогом T_{ij}. Важливо зауважити, що поріг може не бути однаковим для різних частот. Враховуючи що значення високих частот матриці f_{ij} звичайно нижчі ніж значення для низьких частот, поріг T_{ij} має бути меншим.

Конкретне значення порогу T_{ij} залежить від типу зображення над яким проводяться операцій і має бути відкоригований залежно від конкретних реалізацій. Один з кращих способів визначити поріг T_{ij} є статистичний підхід. В першу чергу вибирається тренувальний набір зображень залежно від галузі застосування. Тоді вираховується матриця f_{ij} для кожної точки інтересу у кожному зображенні з тренувального набору. Після цього для кожної пари точок i,j та медіан m_{ij} вираховується f_{ij}. Значення m_{ij} може бути безпосередньо застосовано як поріг.

Після вибірки за пороговим значенням, описаним вище, розмір пам'яті, яку займає дескриптор, зменшується до *N біт, де N* - це кількість не відкинути коефіцієнтів матриці. Для того, щоб покращити визначальність дескриптора за рахунок його розміру значні від'ємні та додатні коефіцієнти матриці можуть зберігатися окремо, таким чином збільшуючи розмір дескриптора до 2*N біт*.

Отриманий таким чином дескриптор області представляє собою бітові рядки, які об’єднуються у один дескриптор цілої площі.

3.3. Алгоритм порівняння дескрипторів характеристик

У дескрипторах заснованих на просторовому представленні зображення, порівняння дескрипторів здійснюється на підставі евклідової відстані за формулою суми квадратів різниць:

$$De(p,q) = [(x-s)^2 + (y-t)^2]^2$$

Де p, q - точки з координатами $(x,y), (s, t)$.

На відміну від дескрипторів SIFT чи SURF, де відстань вираховується як евклідова відстань між векторами дескрипторів, для дескрипторів у частотному діапазоні відстань визначається як відстань Геммінга між двома послідовностями бітів. Таким чином відстань між двома дескрипторами може визначатися як вага Геммінга, що означає побітову операцію XOR з мірою складності O(log n) де n - є довжина бітового рядка. Таким чином досягається значне збільшення швидкості порівняння дескрипторів.

Приклад реалізації відстані Гемінга для бітових рядків мовою Java:

```
int hamming_distance(unsigned x, unsigned y)
{
    int dist = 0;
    unsigned  val = x ^ y;

    // Count the number of bits set
    while (val != 0)
    {
        // A bit is set, so increment the count and clear the
bit
        dist++;
        val &= val - 1;
    }

    // Return the number of differing bits
    return dist;
}
```

4. РОЗРОБКА СИСТЕМИ ПОРІВНЯННЯ ЗОБРАЖЕНЬ

Програмна реалізація описаного у розділі 3 алгоритму має загальну блок-схему, представлену на рис 4.1.

Рисунок 4.1. - загальна блок-схема програмної реалізації.

Робота програми реалізована мовою Java, загальна діаграма класів представлена на рис 4.2.

Початковом роботи програмної реалізації алгоритму порівняння зображень є створення екземпляру класу `BigfootDiff`, який має метод compare з наступною сигнатурою:

```
   public DiffResultImpl compare(BufferedImage image1,
BufferedImage image2)
```

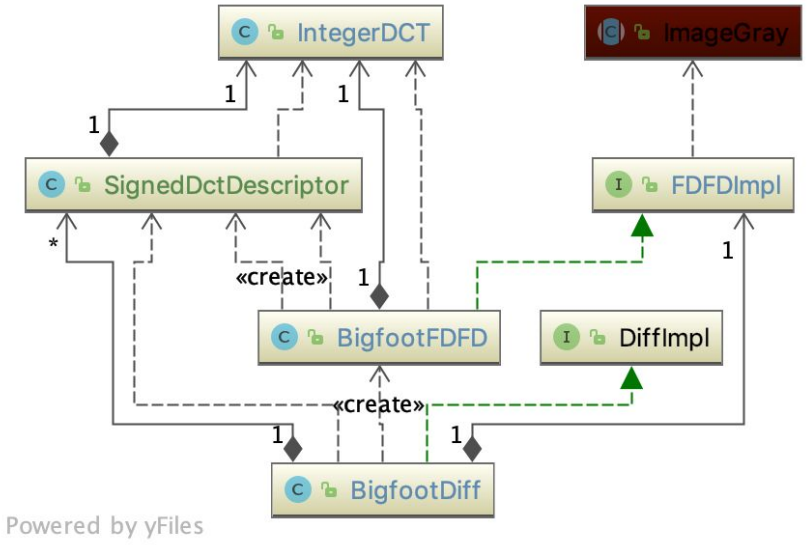

Рисунок 4.2. - діаграма класів програмної реалізації.

На вхід цей метод отримує два зображення для порівняння, а на вихід повертає результат у вигляді об’єкта з інтерфейсом `DiffResultImpl`, конкретна реалізація якого має аксессори до даних про масиви ключових точок для обох зображень, масиви дескрипторів ключових точок, так об’єднуючу структуру для зберігання результатів порівняння `AssociatedIndex`.

4.1. Програмна реалізація виявлення ключових точок

Для виявлення ключових точок на зображеннях, що порівнюються використана бібліотека з відкритим доступом BoofCV, яка розповсюджується за ліцензією Apache 2.0 як для академічних так і для комерційного використання.

У бібліотеці реалізований клас для виявлення ключових точок за алгоритмом SURF, так званий Fast-Hessian.

Далі наведено приклад використання класу для виявлення ключових точок.

```
   // Create a Fast Hessian detector from the SURF paper.
InterestPointDetector<T> detector =
FactoryInterestPoint.fastHessian(new ConfigFastHessian(10, 2,
100, 2, 9, 3, 4));
```

```
// find interest points in the image
detector.detect(input);
// Show the features
displayResults(image, detector);
```

Сигнатура конструктора конфігурації детектора Fast-Hessian:

```
public ConfigFastHessian(float detectThreshold,
                         int extractRadius,
                         int maxFeaturesPerScale,
                         int initialSampleSize,
                         int initialSize,
                         int numberScalesPerOctave,
                         int numberOfOctaves)
```

public float detectThreshold - мінімальна інтенсивність характеристики, залежить від конкретного зображення, рекомендовано налагоджувати під конкретні випадки, починаючи з 1.

public int extractRadius - радіус, який використовується для пригнічення не максимальних значень (NMS), зазвичай 1 або 2.

public int maxFeaturesPerScale - максимальна кількість характеристик, які NMS може повертати на кожному масштабі. Пріорітет надається характеристикам з вищою інтенсивністю. Якщо встановити це значення ≤ 0 NMS поверне усі знайдені характеристики.

public int initialSampleSize - частота вибірки в першій октаві, зазвичай рівна 1 або 2.

public int initialSize - початковий розмір характеристик, зазвичай рівний 9

public int numberScalesPerOctave - кількість октав на рівень масштабу, зазвичай 4

public int numberOfOctaves - кількість октав, зазвичай 4

public int scaleStepSize - різниця у рівнях масштабів. Для деяких наборів даних, встановлення цього значення вищим за заздалегідь встановлене, призводить до покращення стабільності. За замовчуванням рівне 6.

Демонстрація роботи методу зображена на рис 4.3.

Рисунок 4.3. - виявлені ключові точки на зображенні (розмір квадрата навколо точки відповідає масштабу, на якому виявлено точку)

4.2. Програмна реалізація описання ключових точок

Дескриптор ключових реалізований у класі `SignedDctDescriptor`, розширений лістинг коду якого наведено у Додатку 1.

Дискретне косинусне перетворення, на основі якого будується дескриптор реалізоване у класі `IntegerDCT` відповідно до рівняння 3.1. Для застосування формули до зображень створюється матриця перетворень T_{ij}:

$$T_{ij} = \begin{cases} \frac{1}{\sqrt{N}} & \text{if } j = 0 \\ \sqrt{\frac{2}{N}} \cos\left[\frac{(2i+1)j\pi}{2N}\right] & \text{if } j > 0 \end{cases} \tag{4.1}$$

Така матриця створюється одразу при ініціалізації класу `IntegerDCT`:

```
int[][] dct = new int[freqs][freqs];
```

```
        for (int i = 0; i < dct.length; i++) {
            for (int j = 0; j < dct[i].length; j++) {
                dct[i][j] = (int) round((j == 0 ? INV_SQRT_2 :
cos((2 * i + 1) * j * C)) * (1 << (SHIFT - 1)));
            }
        }
```

Коефіцієнти T_{ij} визначаються заздалегідь та зберігаються до таблиці значень (look up table), яка реалізована як властивість `private final int[][] lut;` екземпляру класу `SignedDctDescriptor:`

```
for (int i = 0; i < freqs; i++) {
            for (int j = 0; j < freqs; j++) {
                for (int k = 0; k < dct.length; k++) {
                    for (int l = 0; l < dct.length; l++) {
                        lut[zigzag[i][j]][zigzag[k][l]] =
dct[k][i] * dct[l][j];
                    }
                }
            }
        }
```

Двовимірний масив з n × n коефіцієнтів змінює порядок за допомогою зигзаг-упорядкування (званого також Z-упорядкуванням). Упорядкування зигзагом видно з рис. 4.4., де показана черговість, в якій беруться коефіцієнти.

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Рисунок 4.4. - зигзаг-упорядкування.

Сформований так одновимірний масив з n^2 коефіцієнтів містить довгі серії постійних кодів, у другій половині - нулів, які будуть відкинуті у наступних кроках.

Зигзаг-упорядкування у реалізоване як властивість `zigzag` екземпляру класу `SignedDctDescriptor` і виражене у виді двовимірного масиву:

```
int[][] zigzag = new int[][]{
                {0, 1, 5, 10, 14, 15, 27, 28},
                {2, 4, 6, 12, 16, 26, 29, 42},
                {3, 7, 8, 17, 25, 30, 41, 43},
                {9, 11, 18, 24, 31, 40, 44, 53},
                {13, 19, 23, 32, 39, 45, 52, 54},
                {20, 22, 33, 38, 46, 51, 55, 60},
                {21, 34, 37, 47, 50, 56, 59, 61},
                {35, 36, 48, 49, 57, 58, 62, 63}}
```

```
for (int i = 0; i < freqs; i++) {
            for (int j = 0; j < freqs; j++) {
                for (int k = 0; k < dct.length; k++) {
                    for (int l = 0; l < dct.length; l++) {
                        lut[zigzag[i][j]][zigzag[k][l]] =
dct[k][i] * dct[l][j];
                    }
                }
            }
        }
```

Екземпляр класу `IntegerDCT` створюється після виявлення ключових точок та передається до екземпляру класу `SignedDctDescriptor`, який створюється для кожної ключової точки.

Кожна з ключових точок, знайдених на попередньому кроці передається до цього екземпляру класу `SignedDctDescriptor`. Дескриптор створюється у

методі describe, який використовує інтегральне зображення по зменшення часу на розрахунок сум значень пікселів для DCT.

Дескриптор зберігає описання для трьох регіонів області зображення навколо ключової точки, окремо для всього регіону, центру і границі. Значення описань регіонів зберігаються у наступних властивостях екземпляру класу `SignedDctDescriptor`:

```
    public int centerDesc;
    public long cornerDesc;
    public long borderDesc;
```

Кожне з трьох описань зменшується шляхом відкидання високих частот, також відкидається перший елемент у матриці частотного діапазону, після чого кожне абсолютне значення порівнюється із заздалегідь визначеним порогом, якщо це так, то у відповідну позицію одного біта в одному байті записується одиниця, усі три зазначені вище операції проводяться в одному методі, код якого наведено нижче:

```
private byte positiveDescriptor(int[] dct) {
        byte desc = 0x00;
        for (int i = 0; i < DESCRIPTOR_SIZE; i++) {
            if (dct[i + 1] >= threshold) {
                desc |= (1 << i);
            }
        }
        return desc;
    }
```

4.3. Програмна реалізація порівняння дескрипторів характеристик зображення

Порівняння дескрипторів характеристик зображення основане на відстані Геммінга, яка визначається як числом різних значень для однієї позиції у двох двійкових рядках рис 4.4.

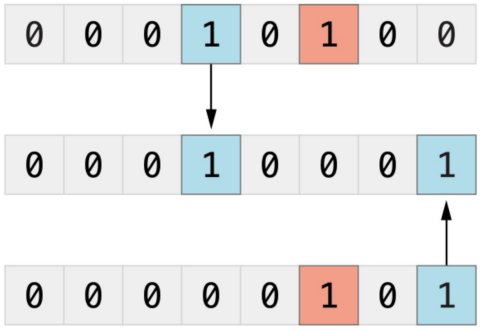

Рисунок 4.4. - визначення відстані Геммінга

Відстань Геммінга визначається бітовою операцією XOR (^).

Розрахунок відстані від одного дескриптора до іншого реалізований у методі `distanceFrom` класу `SignedDctDescriptor наступним чином:`

```
public int distanceFrom(SignedDctDescriptor otherDesc) {
        return Integer.bitCount(centerDesc ^
otherDesc.centerDesc) +
                Long.bitCount(borderDesc ^
otherDesc.borderDesc) +
                Long.bitCount(cornerDesc ^
otherDesc.cornerDesc);
    }
```

Таким чином, чим менша відстань між дескрипторами, тобто чим менше різниця ними, тим більш схожими є ці дескриптори.

Загалом, на підставі кількості схожих дескрипторів робиться висновок про схожість зображень.

4.4. Програмний інтерфейс користувача

Для використання програмного продукту розроблений програмний інтерфейс користувача. Загальна схема роботи користувача з програмним продуктом наведена на рис. 4.5.

Рисунок 4.5. Діаграма взаємодії користувача з програмним продуктом

Для проведення порівняння користувач завантажує ресурс File manager для вибору шляху медіа ресурсів для порівняння. Порівнювати можна не тільки 2 зображення, але й відео ресурси див. рис 4.6.

Рисунок 4.6. - інтерфейс ресурсу File manager

З обраних файлів або папок користувач створює список медіа (media list) та переходить до наступного ресурсу - Job Commander див. рис 4.7.

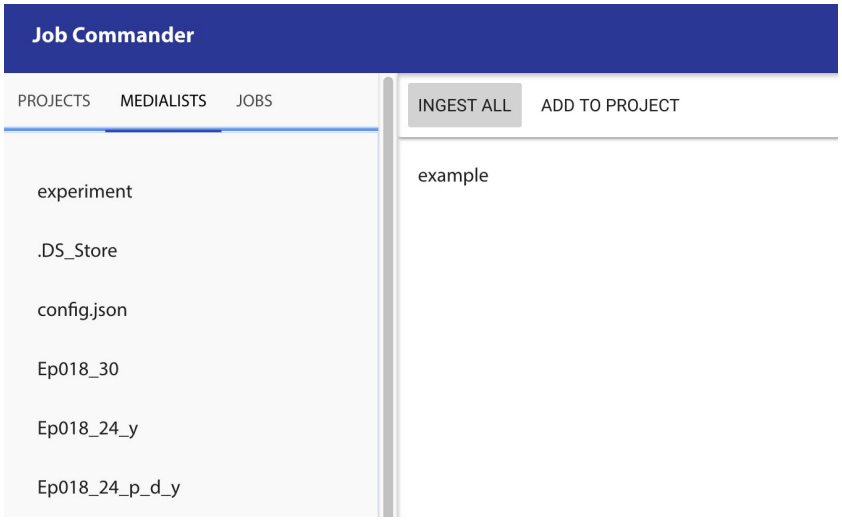

Рисунок 4.7. - інтерфейс управління списками медіа та готовими проектами

Для ініціювання порівняння користувач обирає відповідний список медіа та нажимає кнопку “Ingest All”, переглянути хід виконання порівнянн користувач може перейшовши до вкладки “Jobs”.

Базою для порівняння (master) є перший у списку ресурс, він порівнюється з усіма іншими ресурсами, що є у списку.

Якщо ресурсом є відео, то в результаті порівняння програма видасть список номерів кадрів у базовому відео (master) та номер кадру з порівнюваних відео, який має найкраще співпадіння (найменшу відстань). Поскільки у відео частіше всього співпадають не поодинокі кадри, а сцени, співпадіння об’єднуються у проміжок, та відображаються як одне співпадіння.

Користувач може переглянути результат в інтерфейсі Diff. На рис 4.8. зображено результат порівняння одного відео кліпу з частиною цього ж відео кліпу та показано кадри, де ці два відео співпадають.

Рисунок 4.8. - інтерфейс для перегляду співпадінь у відео

5. ЕКСПЕРИМЕНТАЛЬНА СИСТЕМА ПОРІВНЯННЯ ЗОБРАЖЕНЬ

5.1. Експериментальні результати

Для визначення ефективності розробленого алгоритму проведені порівняльні експерименти. Для проведення експерименту тестова вибірка зображень складається з 25 пар зображень різноманітної тематики (природа, будинки, об’єкти, люди, та ін.), для виявлення ключових точок застосований однаковий алгоритм Fast-Hessian з однаковими налаштуваннями з бібліотеки BoofCV, для порівняння дескрипторів обрано бібліотечну реалізацію алгоритму SURF, критерієм оцінки є точність співвідношення, яке визначається наступним чином:

$$\text{точність} = \frac{\text{правильні співпадіння}}{\text{загальна кількість співпадінь}}$$

Правильні співпадіння визначаються візуальною оцінкою. Для кожного з алгоритмів вибирається відсоток впевненості у розмірі 90%.

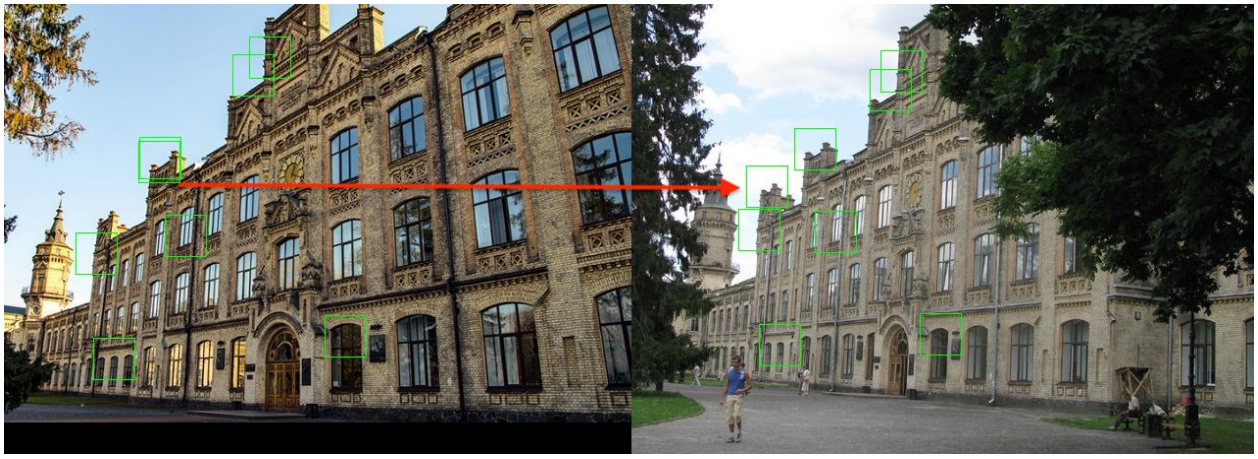

Рисунок 5.1. - приклад неправильного співпадіння виявленого алгоритмом порівняння

Результати проведеного експерименту наведено у таблиці 5.1.

Алгоритм	Точність
SURF	86.8%
Алгоритм дослідження	86.6%

Таблиця 5.1. Порівняння точності співпадінь алгоритмів

Таким чином, видно, що точність алгоритму не поступається галузевому стандарту, оскільки предметом дослідження є алгоритм прискорення, збереження стандартної точності є дуже хорошим показником для алгоритму.

Порівняння середньої швидкості для порівняння 25 пар зображень в тому самому наборі зображень наведена у таблиці 5.2.

Алгоритм	Швидкість мілісекунд/порівняння
SURF	392
Алгоритм дослідження	301

Таблиця 5.2. Порівняння швидкості роботи алгоритму порівняння зображень

Таким чином, ми бачимо покращення показників швидкості на 23%.

За результатами проведеного експерименту можна зробити висновок, що алгоритм порівняння, розроблений в ході цього дослідження показує 23% прискорення для операції порівнянь при збереженні точності порівняння.

6. РОЗРОБКА СТАРТАП ПРОЕКТУ

Назва проекту: “Методи прискорення порівняння зображень”

Комерційна назва: Bigfoot

Опис ідеї стартап-проекту

Таблиця 6.1. - опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Програмне забезпечення для знаходження співпадінь у різних редакціях одного фільму.	1. Пошук вирізаних сцен у фільмі	Швидке визначення сцен у фільмі, що були вирізані, у порівнянні з іншою редакцією
	2. Пошук редакції фільму до якої увійшла сцена	Швидке визначення фільму до якої увійшла сцена
	3. Реконструкція списку монтажних рішень (Edit decision list)	Реконструкція монтажу фільму з вихідних плівок

Визначення сильних, слабких та нейтральних характеристик ідеї проекту

Порівняльний аналіз показників: для власної ідеї визначаються показники, що мають а) гірші значення (W, слабкі); б) аналогічні (N, нейтральні) значення; в) кращі значення (S, сильні)

Таблиця 6.2. - Визначення сильних, слабких та нейтральних характеристик ідеї проекту

N	Техніко-е		W	N	S

о п/п	кономічні характеристики ідеї	(потенційні) товари/концепції конкурентів		(слабка сторона)	(нейтральна сторона)	(сильна сторона)
		Bigfoot	Людський ресурс			
1.	Час порівняння зображень	1-2 доби	2-3 місяці			+
2.	Точність	Задовільна, потребує додаткового уточнення	Не потребує додаткового уточнення	+		
3.	Постійні витрати	Підтримка та оновлення ПО	Витрати на оплату робочого часу		+	
4.	Залежність від рекламодавців	так	ні		+	
5.	Бар’єри проникнення	високі	низькі	+		
6.	Інші джерела доходів	немає	є	+		
7.	Плата за користування	є	немає	+		
8.	Персоналі	є	є		+	

	зація					
9.	Кількість корисних застосувань	Більше одного	один			+

Технологічна здійсненність ідеї проекту

Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових:

за якою технологією буде виготовлено товар згідно ідеї проекту; чи існують такі технології, чи їх потрібно розробити/доробити; чи доступні такі технології авторам проекту;

Таблиця 6.3. - Технологічна здійсненність ідеї проекту

N	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Порівняння зображень засобами комп’ютерного алгоритму	Алгоритм порівняння зображень	існує	доступна
2.	Обробка відео для здійснення порівняння покадрово	Виокремлення кадрів відео у масив зображень	існує	доступна
3.	Обробка	Аналіз	існує	доступна

	зображень для нормалізації	зображення для виявлення гомографії		

Визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи:

- Потреба, що формує ринок - базова потреба, яку задовольняє товар (згідно концепції потенційного товару)
- Цільова аудиторія (цільові сегменти ринку) - визначити потенційні цільові групи клієнтів, що можуть бути зацікавлені у задоволені означеної потреби
- Відмінності у поведінці різних потенційних цільових груп клієнтів - фактори, що формують поведінку клієнта (стандарти, технічні регламенти, інші фактори цінового та нецінового характеру) та особливості купівлі та експлуатації товару

Таблиця 6.4. - Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Економічний	Проект залежить від кіностудій. Невелика кількість клієнтів збільшує залежність від одного-двох партнерів. У разі відмови кіностудій від співробітництва, проект перестає бути рентабельним.	Знайти більш широке застосування для продукту.

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища.

Таблиця 6.5. - SWOT - аналіз стартап-проекту

Сильні сторони: - Відсутність прямих конкурентів - Відсутність фізичних товарів - Необмежена локалізація	Слабкі сторони: - Залежність від партнерів - Висока вартість підтримки програмного комплексу
Можливості: - Вихід на міжнародний рівень - Постійне збільшення кількості користувачів	Загрози: - Відмова партнерів від співробітництва - Копіювання продукту конкурентом

ВИСНОВКИ

В результаті проведеного дослідження можна зробити наступні висновки щодо вирішення задачі прискорення зображень:

1. Представлено алгоритм описання характеристик зображень в частотному діапазоні, на основі дискретного косинусного перетворення, який показує точність презентації визначальних характеристик зображення;
2. Завдяки нормалізації дескриптора характеристик, досягнуто стійкості до змін контрасту та освітлення зображення;
3. Завдяки відкиданню високих частот дескриптора досягнуто зменшення його розміру до 8 частот;
4. Приведення дескриптора до бінарного виду зменшує розмір даних з двовимірного масиву десяткових дробів до *N біт, де N* - це кількість не відкинути коефіцієнтів;
5. Відстань між двома дескрипторами може визначатися як вага Геммінга, що означає побітову операцію XOR з мірою складності O(log n) де *n* - є довжина бітового рядка;
6. В результаті оптимізації розміру даних та часу розрахунку описання характеристик досягнуто прискорення часу порівняння зображень на 23%.

Таким чином, розроблений у ході досліджень алгоритм має теоретичне обґрунтування та експериментальне підтвердження прискорення порівняння зображень зі збереженням точності описання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lowe D. Distinctive image features from scale-invariant keypoints, cascade filtering approach. IJCV №60 – 2004. – p. 91 – 110.
2. Bay H. SURF: Speeded Up Robust Features / Herbert Bay, Tinne Tuytelaars, Luc Van Gool // Computer Vision – ECCV – 2006. – p. 404 – 417.
3. Harris, C. A Combined Corner and Edge Detector. / Chris Harris, Mike Stephens // Paper presented at the meeting of the Proceedings of the 4th Alvey Vision Conference, 1988.
4. Rosten E. Machine Learning for High-Speed Corner Detection. / Rosten Edward, Drummond Tom // Computer Vision – ECCV – 2006. – p. 430 – 443.
5. Lewis J.P. Fast template matching / Lewis J.P. // Proc. Vision Interface – 1995. – p. 120–123.
6. Heinly J. Comparative evaluation of binary features / Heinly J., Dunn E., Frahm J.-M. // Computer Vision–ECCV, Springer – 2012 – p. 759-773.
7. Crow F. Summed-area tables for texture mapping / Crow, Franklin // SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques – 1984 – p. 207–212.
8. Ahmed N. Discrete Cosine Transform / Ahmed, Natarajan, Rao // IEEE Transactions on Computers 23 – 1974 – p. 90-93.
9. Gonzalez R.C. Digital Image Processing / Gonzalez, R.C., Woods, R.E. – 3rd edn, – Pearson Prentice hall, London, 2008 – ISBN: 0-13-168728-x 978-0-13-168728-8.
10. Wilhelm Burger. Digital Image Processing – An Algorithmic Introduction Using Java / Wilhelm Burger and Mark J. Burge. – 2nd Edition, Springer, London, 2016. – 811 p. – ISBN: 978-1-4471-6683-2.
11. Tanimoto, Steven L. An Interdisciplinary Introduction to Image Processing: Pixels, Numbers, and Programs. / Tanimoto, Steven L. – The MIT Press, 2012. – ISBN: 978-0-262-01716-9.

12. Nixon Mark. Feature Extraction and Image Processing. / Nixon Mark – Elsevier Science. 2013. – ISBN: 0-7506-5078-8
13. Davies, E. R. Computer Vision: Principles, Algorithms, Applications, Learning. / Davies, E. R. – Elsevier Science. 2017. – ISBN: 978-0-12-809284-2.
14. Solomon Justin. Numerical Algorithms: Methods for Computer Vision, Machine Learning, and Graphics. / Solomon, Justin – CRC Press. Kindle Edition. 2015 – ISBN:978-1-4822-5189-0.
15. Mikolajczyk K. A performance evaluation of local descriptors / Mikolajczyk K., Schmid C. // Pattern Analysis and Machine Intelligence Vol. 27, Number 10. – 2005 – p. 1615-1630.
16. Luo J. A Comparison of SIFT, PCA-SIFT, and SURF / Luo J., Oubong G. // International Journal of Image Processing (IJIP) 3(4) – 2009. – p.143-152.

Додаток 1. Лістинг коду класа SignedDctDescriptor

```
public class SignedDctDescriptor {
    public int centerDesc;
    public long cornerDesc;
    public long borderDesc;
    private static DescriptorShrinker shrinker = new
DescriptorShrinker(30);
    private transient IntegerDCT dct;
    // storage for kernels used to compute laplacian sign
    protected IntegralKernel kerXX;
    protected IntegralKernel kerYY;

    public SignedDctDescriptor(int x, int y, int size) {
        this.x = (short) x;
        this.y = (short) y;
        this.size = (byte) size;
    }

    public SignedDctDescriptor(ScalePoint p) {
        this(p, IntegerDCT.createDCT8forFDD());
    }

    public SignedDctDescriptor(int x, int y, int size,
IntegerDCT dct) {
        this(x, y, size);
        this.dct = dct;
    }

    public SignedDctDescriptor(ScalePoint p, IntegerDCT dct) {
        this((int) round(p.x), (int) round(p.y), (int)
round(p.scale * DESC_REGION_SIZE));
        this.dct = dct;
    }

    public int distanceFrom(SignedDctDescriptor otherDesc) {
```

```
        return Integer.bitCount(centerDesc ^
otherDesc.centerDesc) +
                Long.bitCount(borderDesc ^
otherDesc.borderDesc) +
                Long.bitCount(cornerDesc ^
otherDesc.cornerDesc);
    }

    public SignedDctDescriptor describe(GrayS32 integralImage)
{
        ImageRegion[] regions = getRegions(integralImage);

        // descriptor of the whole feature with surroundings
        ImageRegion region = regions[WHOLE];
        centerDesc = describe(region, WHOLE);
        // descriptor of the feature itself (center region)
        centerDesc |= describe(regions[CENTER], CENTER) << 16;

        // descriptor of the feature surroundings:
        borderDesc = (describe(regions[TOPLEFT], TOPLEFT) |
(describe(regions[TOPRIGHT], TOPRIGHT) << 16)) &
0x00000000ffffffffL;
        borderDesc <<= 32;
        borderDesc |= (describe(regions[BOTTOMRIGHT],
BOTTOMRIGHT) | (describe(regions[BOTTOMLEFT], BOTTOMLEFT) <<
16)) & 0x00000000ffffffffL;
        cornerDesc = (describe(regions[TOP], TOP) |
(describe(regions[RIGHT], RIGHT) << 16)) &
0x00000000ffffffffL;
        cornerDesc <<= 32;
        cornerDesc |= (describe(regions[BOTTOM], BOTTOM) |
(describe(regions[LEFT], LEFT) << 16)) & 0x00000000ffffffffL;

        computeLaplaceSign(integralImage);

        return this;
```

```
    }

    public int describe(ImageRegion region, int area) {
        int[] dct = this.dct.dct(region);
        int[] normalized =
DescriptorArea.values()[area].normalizer.normalize(dct);
        return shrinker.signedDescriptor(normalized) & 0xffff;
    }

    public static int size() {
        return 26; // 2 + 2 + 1 + 1 + 4 + 8 + 8
    }

    public static int size(SignedDctDescriptor... descs) {
        return size() * descs.length;
    }

    public int sizeof() {
        return size();
    }

    public byte[] toBytes() {
        byte bytes[] = new byte[sizeof()];
        ByteBuffer bb = ByteBuffer.wrap(bytes);
        write(bb);
        return bytes;
    }

    public ByteBuffer write(ByteBuffer bb) {
        bb.putShort(x);
        bb.putShort(y);
        bb.put(laplacian ? (byte) 1 : 0);
        bb.put(size);
        bb.putInt(centerDesc);
        bb.putLong(borderDesc);
        bb.putLong(cornerDesc);
```

```
        return bb;
    }

    public short getAreaDescriptor(DescriptorArea area) {
        switch (area) {
            case WHOLE:
                return (short) (centerDesc & 0xffff);
            case CENTER:
                return (short) ((centerDesc >> 16) & 0xffff);
            case BOTTOMRIGHT:
                return (short) (borderDesc & 0xffff);
            case BOTTOMLEFT:
                return (short) ((borderDesc >> 16) & 0xffff);
            case TOPLEFT:
                return (short) ((borderDesc >> 32) & 0xffff);
            case TOPRIGHT:
                return (short) ((borderDesc >> 48) & 0xffff);
            case BOTTOM:
                return (short) (cornerDesc & 0xffff);
            case LEFT:
                return (short) ((cornerDesc >> 16) & 0xffff);
            case TOP:
                return (short) ((cornerDesc >> 32) & 0xffff);
            case RIGHT:
                return (short) ((cornerDesc >> 48) & 0xffff);
            default:
                throw new IllegalArgumentException("Unknown
area " + area);
        }
    }

    public String getAreaDescriptorString(DescriptorArea area)
{
        short areaDesc = getAreaDescriptor(area);

        StringBuilder sb = new StringBuilder();
```

```
        for (int i = 0; i < 8; i++) {
            if ((areaDesc >> i + 8) % 2 == 1) {
                sb.append("-");
            } else if ((areaDesc >> i) % 2 == 1) {
                sb.append("+");
            } else {
                sb.append("0");
            }
        }
        return sb.toString();
    }

    @Override
    public String toString() {
        StringBuilder sb = new
StringBuilder(String.format("Descriptor for point (%s, %s),
scale %s:\n", x, y, size));
        for (DescriptorArea area : DescriptorArea.values()) {

sb.append("\t").append(area).append(area.toString().length() <
7 ? "\t:\t" :
":\t").append(getAreaDescriptorString(area)).append("\n");
        }
        return sb.toString();
    }

    public void read(ByteBuffer bb) {
        x = bb.getShort();
        y = bb.getShort();
        laplacian = bb.get() == 1;
        size = bb.get();
        centerDesc = bb.getInt();
        borderDesc = bb.getLong();
        cornerDesc = bb.getLong();
    }
```

```
    public void fromBytes(byte[] bytes) {
        read(ByteBuffer.wrap(bytes));
    }

    @Override
    public boolean equals(Object obj) {
        if (obj.getClass() != this.getClass()) return false;
        SignedDctDescriptor other = (SignedDctDescriptor) obj;
        return new EqualsBuilder().append(x,
other.x).append(y, other.y)
                .append(size, other.size).append(laplacian,
other.laplacian)
                .append(cornerDesc,
other.cornerDesc).append(centerDesc, other.centerDesc)
                .append(borderDesc, other.borderDesc).build();
    }

    @Override
    public int hashCode() {
        return new
HashCodeBuilder().append(x).append(y).append(size).append(lapl
acian)

.append(cornerDesc).append(centerDesc).append(borderDesc).buil
d();
    }


    protected byte size;
    protected short x;
    protected short y;
    public boolean laplacian;

    public static final int DESC_REGION_SIZE = 8;
    final static int WHOLE = 0;
    final static int TOPLEFT = 1;
```

```
    final static int TOP = 2;
    final static int TOPRIGHT = 3;
    final static int LEFT = 4;
    final static int CENTER = 5;
    final static int RIGHT = 6;
    final static int BOTTOMLEFT = 7;
    final static int BOTTOM = 8;
    final static int BOTTOMRIGHT = 9;

    protected void computeLaplaceSign(GrayS32 ii) {
        kerXX =
DerivativeIntegralImage.kernelDerivXX(getSize(), kerXX);
        kerYY =
DerivativeIntegralImage.kernelDerivYY(getSize(), kerYY);
        double lap = convolveSparse(ii, kerXX, x, y) +
convolveSparse(ii, kerYY, x, y);

        laplacian = lap > 0;
    }

    protected ImageRegion[] getRegions(GrayS32 ii) {
        ImageRegion[] regions = new ImageRegion[10];
        regions[WHOLE] = new ImageRegion(ii, x, y, size);
        int subregionSize = round(size / 3f);
        regions[TOPLEFT] = new ImageRegion(ii, x -
subregionSize, y - subregionSize, subregionSize);
        regions[TOP] = new ImageRegion(ii, x, y -
subregionSize, subregionSize);
        regions[TOPRIGHT] = new ImageRegion(ii, x +
subregionSize, y - subregionSize, subregionSize);
        regions[LEFT] = new ImageRegion(ii, x - subregionSize,
y, subregionSize);
        regions[CENTER] = new ImageRegion(ii, x, y,
subregionSize);
        regions[RIGHT] = new ImageRegion(ii, x +
subregionSize, y, subregionSize);
```

```
        regions[BOTTOMLEFT] = new ImageRegion(ii, x -
subregionSize, y + subregionSize, subregionSize);
        regions[BOTTOM] = new ImageRegion(ii, x, y +
subregionSize, subregionSize);
        regions[BOTTOMRIGHT] = new ImageRegion(ii, x +
subregionSize, y + subregionSize, subregionSize);
        return regions;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    public int getSize() {
        return size & 0xff;
    }

    public boolean hasPositiveLaplacian() {
        return laplacian;
    }

    public static int convolveSparse(GrayS32 integral,
IntegralKernel kernel, int x, int y) {
        int ret = 0;
        int N = kernel.getNumBlocks();

        for (int i = 0; i < N; i++) {
            ImageRectangle r = kernel.blocks[i];
            ret += block_zero(integral, x + r.x0, y + r.y0, x
+ r.x1, y + r.y1) * kernel.scales[i];
        }
```

```
        return ret;
    }

    public static int block_zero(GrayS32 integral, int x0, int
y0, int x1, int y1) {
        x0 = min(x0, integral.width - 1);
        y0 = min(y0, integral.height - 1);
        x1 = min(x1, integral.width - 1);
        y1 = min(y1, integral.height - 1);

        int br = 0, tr = 0, bl = 0, tl = 0;

        if (x1 >= 0 && y1 >= 0)
            br = integral.data[integral.startIndex + y1 *
integral.stride + x1];
        if (y0 >= 0 && x1 >= 0)
            tr = integral.data[integral.startIndex + y0 *
integral.stride + x1];
        if (x0 >= 0 && y1 >= 0)
            bl = integral.data[integral.startIndex + y1 *
integral.stride + x0];
        if (x0 >= 0 && y0 >= 0)
            tl = integral.data[integral.startIndex + y0 *
integral.stride + x0];

        return br - tr - bl + tl;
    }
}
```

Додаток 2. Публікція